The Expressive Power of Word Embeddings

Yanqing Chen[*]
Computer Science Dept.
Stony Brook University
cyanqing@cs.stonybrook.edu

Bryan Perozzi[*]
Computer Science Dept.
Stony Brook University
bperozzi@cs.stonybrook.edu

Rami Al-Rfou'[*]
Computer Science Dept.
Stony Brook University
ralfrou@cs.stonybrook.edu

Steven Skiena
Computer Science Dept.
Stony Brook University
skiena@cs.stonybrook.edu

Abstract

We seek to better understand the difference in quality of the several publicly released embeddings. We propose several tasks that help to distinguish the characteristics of different embeddings. Our evaluation of sentiment polarity and synonym/antonym relations shows that embeddings are able to capture surprisingly nuanced semantics even in the absence of sentence structure. Moreover, benchmarking the embeddings shows great variance in quality and characteristics of the semantics captured by the tested embeddings. Finally, we show the impact of varying the number of dimensions and the resolution of each dimension on the effective useful features captured by the embedding space. Our contributions highlight the importance of embeddings for NLP tasks and the effect of their quality on the final results.

1 Introduction

Distributed word representations (embeddings) capture semantic and syntactic features of words out of raw text corpus without human intervention or language dependent processing. Embeddings are a promising model to fight sparsity of the data and push supervised and semi-supervised tasks performance. The features they capture are task independent which make them ideal for language modeling. However, embeddings are hard to interpret and understand. Despite the efforts of visualizing the word embeddings [16], points in high dimensional spaces carry a lot of information that is hard to quantify. Additionally, there is not yet an understanding about the best way to approach learning these representations. Publicly available embeddings have been generated by multiple research groups using different data and training procedures.

We investigate the different characteristics of three different approaches to generate word embeddings: (1) HLBL, (2) SENNA, and (3) Turian's. HLBL uses a log-linear loss function to speed up the training. The prediction of the next word is divided into a sequence of partial predictions that rely on the context history. SENNA and Turian's embeddings both use the hinge loss function to score the corrupted phrase higher than the ones observed in the text. However they differ in how negative training examples are generated. Turian corrupts phrases by replacing the last word with a random one, while SENNA randomizes the word in the middle of the phrase.

[1] *Contributed equally to this work.*

To better understand the variety of semantic meanings captured by word embeddings, we evaluate each in a variety of term classification tasks. The classification tasks aim to test different aspects of the semantics captured by the embeddings. We use term classification rather than sequence labeling tasks (such as part of speech tagging) to isolate the effects of context in making decisions and eliminate the complexity of the learning methods.

Specifically, our work makes the following contributions:

- We show through evaluation that embeddings are able to capture semantics in the absence of sentence structure and that there is a difference in the characteristics of the publicly released word embeddings.
- We explore the impact of the number of dimensions and the resolution of each dimension on the quality of the information that can be encoded in the embeddings space. That shows that minimum effective space needed to capture the useful information in the embeddings.
- We demonstrate the importance of word pair orientation in encoding useful linguistic information. We run two pair classification tasks and provide an example with one of them where pair performance greatly exceeds that of individual words.

The rest of the work proceeds as follows: First we describe the word embeddings we consider. Next we discuss our classification experiments, and present their results. Finally we discuss the effects of scaling down the size of the embeddings space.

2 Related Work

The original work for generating word embeddings was presented by Bengio et. al. in [1]. They generated embeddings by training a language model on a huge amount of text. The embeddings were a secondary output of this time-intensive process (its intent was to generate a language model). Since [1], there has been a significant interest in speeding up the generation process [2, 3]. These original language models were evaluated using perplexity. We argue here that while perplexity is a good metric of language modeling, it is not insightful about how well the embeddings capture diverse types of information. Our work is different in that we propose several tasks for evaluation rather than using one number to summarize quality.

There has been recent interest in the application of embeddings for learning features and representations. SENNA's embeddings [5] are generated using a model that is discriminating and non-probabilistic. In each training update, we read an n-gram $x = (w_1, \ldots, w_n)$ from the corpus, concatenating the learned embeddings of the n words $e(w_1) \oplus \ldots \oplus e(w_n)$ where e is the lookup table and $\oplus$ is concatenation. Then a corrupted n-gram x' is used by replacing the word in the middle with a random one from the vocabulary. On top of the two phrases, the model learns a scoring function S that scores the original phrases lower than the corrupted one. The loss function used for training is hinge loss $L(x) = max(0; 1 - S(x) + S(x'))$. SENNA [6] shows that embeddings are able to perform well on several NLP tasks in the absence of any other features. The NLP tasks considered by SENNA all consist of sequence labeling. This makes it hard to isolate what the model learns from sequence dependencies versus what the embeddings themselves carry as intrinsic information. By focusing on term classification problems, our work enriches the discussion of distributed word representations.

In [15], Turian et. al. duplicated SENNA embeddings with some differences; they corrupt the last word of each n-gram instead of the word in the middle. They also show that using embeddings in conjunction with typical NLP features improves the performance on the Named Entity Recognition task. An additional result of [15] shows that most of the embeddings have similar effect when added to an existing NLP task. This gives the wrong impression - not all embeddings are created equal. Our work illustrates that significant differences in the information captured by each publicly released model exist.

Mnih and Hinton [11] proposed a log-bilinear loss function to model language. Given an n-gram, the model concatenates the embeddings of the n-1 first words, and learns a linear model to predict the embedding of the last word. Mnih and Hinton later proposed Hierarchical log-bilinear (HLBL) model embeddings [12] to speed up model evaluation during training and testing by using a hierarchical approach (similiar to [13]) that prune the search space for the next word by dividing the prediction into a series of predictions that filter region of the space. The language model eventually is evaluate using perplexity.

A fundamental challenge for neural language models involves representing words which have multiple meanings. In [9], Huang et. al. incorporate global context to deal with challenges raised by words with multiple meanings.

3 Experimental setup

In this paper, we will construct three term classification problems and two pair classification problems to quantify the quality of the embeddings. In this section, we discuss the specifics of our tasks and the embeddings.

3.1 Evaluation Tasks

Our evaluation tasks are as follows:

- **Sentiment Polarity**: We use Lydia's sentiment lexicon [8] to create sets of words which have positive or negative connotations and construct the 2-class sentiment polarity test. We also consider a 3-class version of the sentiment test, in which we discriminate between words that are positive, negative, and neutral. We pick our set of neutral words by randomly selecting from words not occurring in our sentiment lexicon.
- **Noun Gender**: We use Bergsma's dataset [4] to compile a list of masculine and feminine proper nouns. Names that corefer more frequently with *she*/*he* are respectively considered feminine/masculine. We ignore the strings that corefer the most with *it*, appear less than 300 times in the corpus, or consist of multiple words.
- **Plurality**: We use WordNet [7] to extract nouns in their singular and plural forms. While this task is not hard to be coded using morphological based rules, the automatic discovery of such features could be beneficial to other languages where singulars are hard to distinguish with rules from singulars.
- **Synonyms and Antonyms**: We use WordNet to extract synonym and antonym pairs and check whether we can part one kind from the others. The relation is a symmetric one. If a is antonym of b, then b is an antonym of a. For instance, *good* is an antonym of *evil* thus *evil* is also an antonym of *good*. To preserve symmetry, for each pair of synonyms and antonyms we will feed the classifier two problems to classify, (a, b) and (b, a). The feature vector for each of them will consist of the concatenation of both word embeddings. We also consider a 3-class version of this test which adds a new group of word relations - those that are neither synonyms nor antonyms.
- **Regional Spellings**: We collect the words that differ in spelling between UK English and the American counterpart from an online source [10]. Even thought this task could be a term classification task, we consider it a pair classification task. We show later that this decision improves the accuracy dramatically. This task is not symmetric as the previous one. Hence, we give two different labels for the pair and its transpose.

We ensure that for all tasks the class labels are balanced. This allow our baseline evaluation to be either the random classifier or the most frequent label classifier. Either of them will give an accuracy of 50% for 2-class-test and 33% for 3-class-test. Table 1 shows examples of each of the 2-class evaluation tasks. In each of them the classifier is asked to identify which of the classes the term or pair belongs to.

	Sentiment		Noun Gender		Plurality	
	Positive	Negative	Feminine	Masculine	Plural	Singular
Samples	good	bad	Ada	Steve	cats	cat
	talent	stupid	Irena	Roland	tables	table
	amazing	flaw	Linda	Leonardo	systems	system

	Synonyms and Antonyms		Regional Spellings	
	Synonyms	Antonyms	UK	US
Samples	store shop	rear front	colour	color
	virgin pure	polite impolite	driveable	drivable
	permit license	friend foe	smash-up	smashup

Table 1: Example input from each task

3.2 Embeddings' Datasets

We choose the following publicly available embeddings datasets for evaluation.

- **SENNA's embeddings** covers 130,000 words with 50 dimensions for each word. They were trained on English Wikipedia articles over weeks.
- **Turian's embeddings** covers 268,810 words, each represented either with 25, 50 or 100 dimensions. To train their embeddings, they used the RCV1 corpus, which contains one year of Reuters English newswire, from August 1996 to August 1997, about 63 millions words in 3.3 million sentences.
- **HLBL's embeddings** covers 246,122 words. These embeddings were trained on same data used for Turian embedding for 100 epochs (7 days), and have been induced in 50 or 100 dimensions.
- **Huang's embeddings** covers 100,232 words, in 50 dimensions. They were induced by training on Wikipedia. Huang's embeddings require context to disambiguate which prototype to use for a word. Our tasks are context free, and so we average the multiple prototypes to a single point in the space. (This was the approach which worked best in our testing.)

It should be emphasized that each of these models has been induced under substantially different training parameters. Each model has its own vocabulary, used a different context size, and was trained for a different number of epochs on its training set.

While the control of these variables is outside the scope of this study, we hope to mitigate one of these challenges by running our experiments on the vocabulary shared by all these embeddings. The size of this shared vocabulary is 58,411 words.

3.3 Classification

For classification we use Logistic Regression, a SVM with a Linear kernel, and a SVM with the RBF-kernel as classifiers. All experiments were written using the Python machine learning package Scikit-Learn [14]. For the term classification tasks we offered the classifier only the embedding of the word as an input.

For the synonyms and antonyms and the regional spellings experiments, the input consists of the embeddings of the two words concatenated. To eliminate any asymmetric bias, our dataset contains each pair with its inverted version.

The average of four folds of cross validation is used to evaluate the performance of each classifier on each task. In each setup, 50%, 25%, 25% of the data are used, as training, development and testing datasets respectiviely, for evaluation and model selection. Model selection is done by executing a grid-search on the parameter space with the help of the development data.

4 Evaluation Results

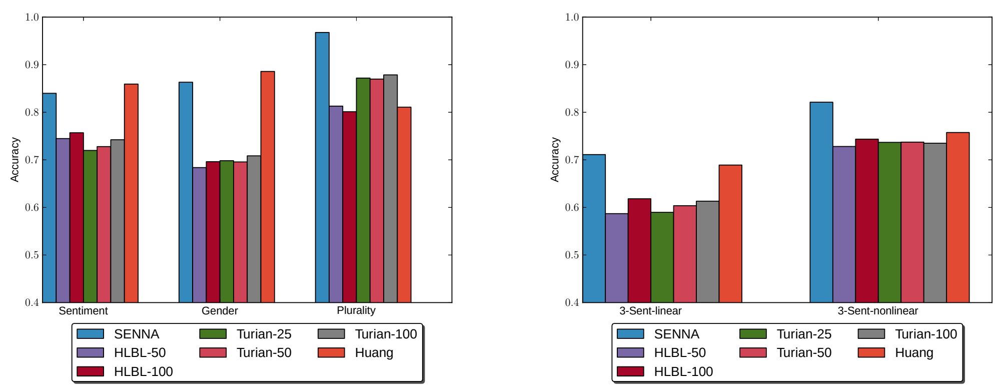
The embeddings are a mapping of words to points in a vector space. The assumption is that the coordinates of the points convey useful information. However, any subset of

dimensions could contribute to any concept and any concept could be represented by multiple dimensions. It is therefore not only hard to interpret the meaning of the coordinates but also to evaluate the correctness of the mapping itself. In this section we present the evaluation of both our term and pair classification results.

4.1 Term Classification

Figure 1a shows the results over all the 2-class term classification tasks, averaging the accuracy the three classifiers with the geometric mean. There are two notable observations to be made about these results. The first is that all the embeddings we considered did much better than the baseline, even on a seemingly hard tests like sentiment detection. This shows the power that embeddings have. The second is that there is strong performance from both the SENNA and Huang embeddings. An interesting difference between the two is that the SENNA embeddings seem to capture the plurality relationship better. This may be from the emphasis that the SENNA embeddings place on shallow syntactic features.

To strengthen these results, we performed a 3-class version of the sentiment test, in which we evaluated the ability to classify words as having positive, negative, or neutral sentiment value. The results are presented in Figure 1b. The results are consistent with those from our 2-label test, and all embeddings perform much higher than the baseline score of 33%. In order to show that embeddings can still perform quite well on this task, we have reported the nonlinear classifier separately from the linear ones.



(a) 2-class term tasks

(b) 3-class sentiment task

Figure 1: Results of the term-based tasks considered. Figure 1a averages results from the 2-class tasks across classifiers using the geometric mean. Figure 1b contains the performance on the 3-class version of the sentiment task. To illustrate that strong performance is still possible on this task, we report results by classifer type seperately.

Table 2 shows examples of words from the test datasets after classifying them using logistic regression on the SENNA embeddings. The top and bottom rows show the words that the classifier is confident classifying, while the rows in the middle show the words that lie close to the decision boundary. For example, *resilient* could have positive and negative connotations in text, therefore, we find it close to the region were the words are more neutral than being polarized.

For SENNA, the best performing task was the Plurality task. That explains the obvious contrast between the probabilities given to the words. The top words are given almost 100% probability and the bottom ones are given almost 0%. The results of regional spelling task is shown here in the term-wise setup. Despite not performing as well as the pair-wise spelling, we can see that classifier shows meaningful results. We can clearly notice that the British spellings of words favor the usage of hyphens, s over z and ll over l.

Sentiment	Positive	Prob	Regional Spelling	British	Prob	Plurality	Plural	Prob
	world-famous	99.85		kick-off	92.37		grantors	99.99
	award-winning	99.83		hauliers	91.54		gainers	99.99
	high-quality	99.83		re-exported	89.46		heifers	99.99
	achievement	99.81		bullet-proof	88.69		Gambians	99.99
	athletic	99.81		initialled	88.42		crushings	99.99
	resilient	50.14		paralysed	50.16		cay	50.29
	ragged	50.11		italicized	50.04		iv	50.12
	discriminating	50.10		exorcise	50.03		leones	50.11
	stout	49.97		fusing	49.90		profanity	49.95
	lose	49.83		lacklustre	49.78		iss	49.81
	bored	49.81		subsidizing	49.77		secrets	49.74
	bloodshed	0.74		signaling	32.04		motion	0.02
	burglary	0.68		hemorrhagic	21.69		wave	0.02
	robbery	0.58		tumor	21.69		tributary	0.02
	panic	0.45		homologue	19.53		by-product	0.02
	stone-throwing	0.28		localize	17.50		clone	0.01
	Negative	1.0-Prob		**American**	1.0-Prob		**Singular**	1.0-Prob

Table 2: Examples of the results of the logistic regression classifier on different tasks.

4.2 Pair Classification

Section 4.1 showed the power of word embeddings in conveying useful features of individual words. Sometimes however, the choice to use pair classification can make quite a difference in the results. Figure 2a shows that classifying individual words according to their regional usage performs poorly. We can redefine the problem such that the classifier is asked to decide if the first word, in a pair of words, is the American spelling or not. Figure 2a shows that performance improves a lot. This hints that the words under this criteria are not separable by a hyper-plane in any subspace of the original embeddings space. Instead, the pairs' positions relative to each other is what encodes such information and not their absolute coordinates.

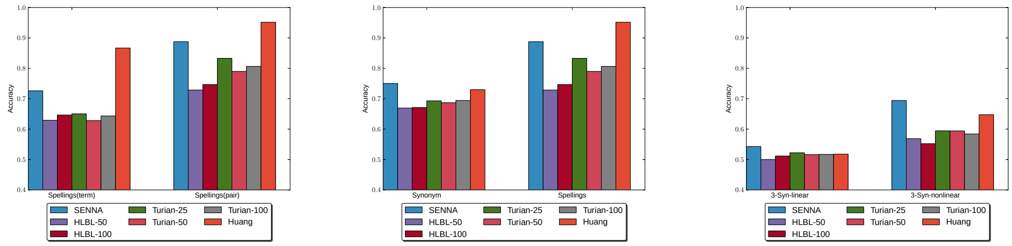
In order to show what forms of linguistic information is encoded in the relative positions between words, we present the results of our 2-class pair tasks in Figure 2b. As before, the embeddings perform well on the tasks and SENNA, in particular, performs best.

We note that it is surprising that neural language models may capture the relation between a synonym and antonym. Both the language modeling of HLBL and the way that SENNA/Turian corrupted their examples favor words that can syntactically replace each other; e.g. *bad* can replace *good* as easily as *excellent* can. The result of this syntactic interchangeability is that both *bad* and *excellent* are close to *good* in the embedding space.

In order to investigate the depth to which synonyms and antonyms are captured, we conducted a 3-class version of the same test. We now evaluate between pairs of words that are synonyms, antonyms, or have no such relation. While such a task is much harder for the embeddings, the results in Figure 2c show that a nonlinear classifier can capture the relationship, particularly with the SENNA embeddings. An analysis of the confusion matrix for the nonlinear SVM showed that errors occurred roughly evenly between the classes. We believe that this finding regarding the encoding of synonym/antonym relationships is an interesting contribution of our work.

5 Information reduction

Distributed word representation exist in continuous space, which is quite different from common language modeling techniques. Beside the powerful expressiveness that we demonstrated previously, another key advantage of distributed representations is their size - they require far less memory and disk storage than other techniques. In this section we seek to understand exactly how much space word embeddings need in order to serve as useful features. We also investigate whether the powerful representation that embeddings offer is a result of having real value coordinates or the exponential number of regions which can be described using multiple independent dimensions.



(a) UK/US term vs. pair (b) 2-class pair results (c) 3-class synonym task

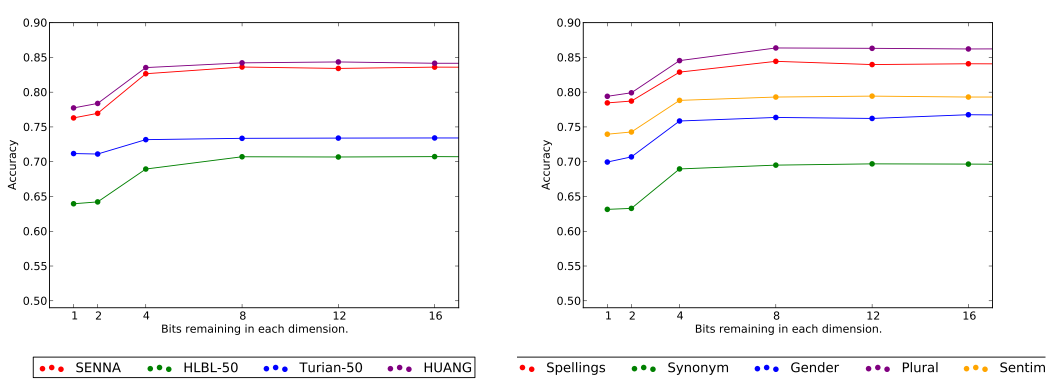
Figure 2: Results of the pair-based tests. Figure 2a shows the difference between treating the UK/US spellings as a single word problem, or using a pair of embeddings. Figure 2b shows the results of the 2-class pair tests together. Both Figures 2a and 2b average their results across classifiers using the geometric mean. Figure 2c shows the performance of the 3-class synonym/antonym task by classifier type.

To understand the effect of such hyper-parameters we run two experiments. The first reduces the resolution of each real-valued dimension and helps us understand the level of precision required for our tasks. The second reduces the dimensions of embeddings and provides insight into how the dimensions of the embeddings effects the final result.

5.1 Bitwise Truncation

To reduce the resolution of the real numbers that make up the embeddings matrix. First we scale them to 32 bit integer values, then we divide the values by 2^b, where b is the number of bits we wish to remove. Finally, we scale the values back to lie between $(-1, 1)$. After this preprocessing we give the new values as features to our classifiers. In the extreme case, when we truncate 31 bits, the values will be all either $\{1, -1\}$.

Figure 3a shows that when we remove 31 bits (i.e, values are $\{1, -1\}$), the performance of an embedding dataset drops no more than 5%. This reduced resolution is equivalent to 2^{50} regions which can be encoded in the new space. This is still a huge resolution, but surprisingly seems to be sufficient at solving the tasks we proposed. A naïve approximation of this trick which may be of interest is to simply take the the sign of the embedding values as the representation of the embeddings themselves.



(a) By embedding (b) By task

Figure 3: Results of reducing the precision of the embeddings, averaged by the geometric mean of classifiers across embeddings (3a) and tasks (3b). We note that after removing 31 bits, each dimension of the embeddings is a binary feature.

5.2 Principle Component Analysis

The bitwise truncation experiment indicates that the number of dimensions could be a key factor into the performance of the embeddings. To experiment on this further, we run PCA over the embeddings datasets to evaluate task performance on a reduced number of dimensions.

Figure 4 shows that reducing the dimensions drops the accuracy of the classifiers significantly across all embedding datasets and all tasks. Looking at Figure 4b, reducing the words embeddings to points on a real line almost deletes all the features that are relevant to the pair classification and to less a degree the sentiment features. Despite the 10%-20% drop in accuracy in the Plurality and Gender tasks, the classification is still higher than random.

The results show that when that shallow syntactic features such as gender and number agreement are preserved at the expense of more subtle semantic features such as sentiment polarity. This gives us insight into what the hierarchical structure of the embeddings space looks like. Shallow semantic features are present in all aspects of the space, and when PCA chooses to maximize this variance of the feature space it is at the expense of the other semantic properties.

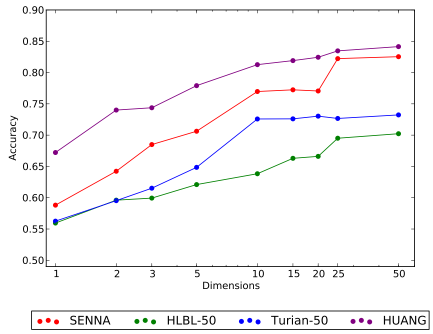
Another key difference between the truncation experiment and the PCA experiment is that the truncation experiment may preserve relationships captured by non-linearities in the embedding space. Linear PCA can not offer such guarantees and this weakness may contribute to the difference in performance. We illustrate this phenomenon in Figure 4c, by showing how the performance of the linear and non-linear classifiers converge for our harder tasks (sentiment and synonym) as we reduce the number of dimensions with PCA.

6 Conclusion

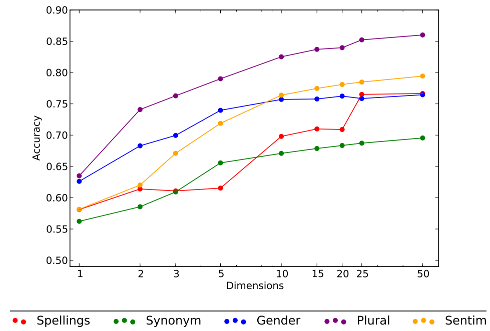
Distributed word representations show a lot of promise to improve supervised learning and semi-supervised learning. The practical advantages of having dense representations make them ideal for industrial applications and software development. The previous work mainly focused on speeding up the training process with one metric for evaluation, perplexity. We show that this metric is not able to convey the features that the embeddings have, or provide a nuanced view of their quality. We develop a suite of linguistic oriented tasks which might serve as a part of a comprehensive benchmark for word embedding evaluation. The tasks focus on words or pairs of them in isolation to the actual text. The goal here is not to build a useful classifier as much as it is to understand how much supervised learning can benefit from the features which are encoded in the embeddings.

We succeed in showing that the publicly available datasets differ in their quality and usefulness, and our results are consistent across tasks and classifiers. Our future work will try to address the factors that lead to such diverse quality. The effect of training corpus size and the choice of the objective functions are two main areas where better understanding is needed.

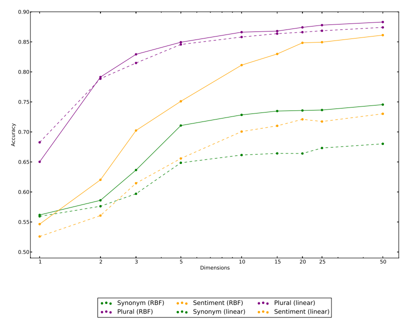
While our tasks are simple, the differences among task performance shed light on the features encoded by embeddings. We showed that in addition to the shallow syntactic features like plural and gender agreement, there are significant semantic partitions regarding sentiment and synonym/antonym meaning. Our current tasks focus on nouns and adjectives, and the suite of tasks has to be extended to include tasks that address verbs and other parts of speech.



(a) By embedding



(b) By task



(c) Linear vs. Nonlinear

Figure 4: Results of reducing the dimensions of the embeddings through PCA, averaged by the geometric mean across embeddings (4a) and task (4b). Figure 4c shows the difference between linear (dashed) and non-linear (solid) classifiers for our harder tasks (sentiment and synonym) and an easy task (plural). The performance of the linear and nonlinear classifiers converges as PCA removes more dimensions. This results in significantly degraded performance on nuanced tasks like sentiment analysis.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[2] Y. Bengio, J.S. Senécal, et al. Quick training of probabilistic neural nets by importance sampling. In *AISTATS Conference*, 2003.

[3] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[4] Shane Bergsma and Dekang Lin. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 33–40, Sydney, Australia, July 2006. Association for Computational Linguistics.

[5] R. Collobert. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[6] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011, JMLR. org.

[7] C. Fellbaum. Wordnet. *Theory and Applications of Ontology: Computer Applications*, pages 231–243, 2010, Springer.

[8] N. Godbole, M. Srinivasaiah, and S. Skiena. Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, volume 2, 2007.

[9] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=2390524.2390645`.

[10] Words Worldwide Limited. Word list of us/uk spelling variants, May 2009. URL `http://www.wordsworldwide.co.uk/docs/Words-Worldwide-Word-list-UK-US-2009.doc`.

[11] A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM, 2007.

[12] A. Mnih and G.E. Hinton. A scalable hierarchical distributed language model. *Advances in neural information processing systems*, 21:1081–1088, 2009, Citeseer.

[13] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252, 2005.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. *Urbana*, 51:61801, 2010.

[16] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.